PRÁCTICA 5. PROGRAMACIÓN MODULAR

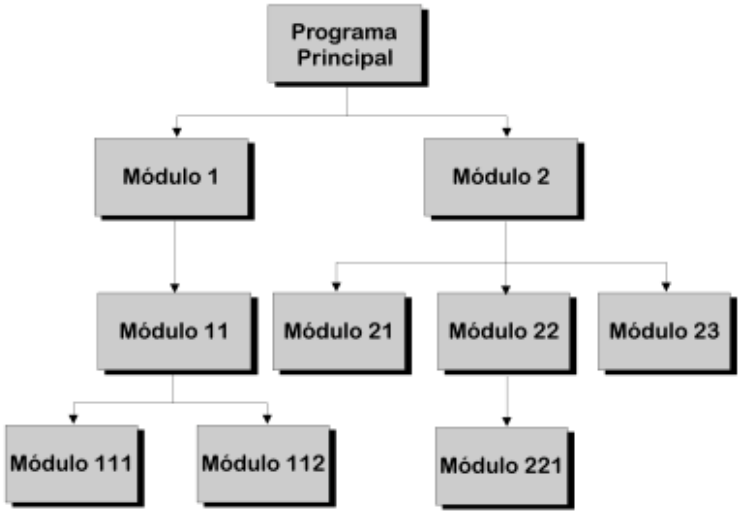
Procedimientos y Funciones.
Paso de Parámetros

2 sesiones Semanas: 21 y 28 de Octubre

OBJETIVOS:

- ✓ Comprender el concepto de módulo
- ✓ Distinguir entre procedimientos y funciones
- ✓ Diferenciar entre paso de parámetros por valor y por referencia
- ✓ Comprender el concepto de retorno de valores en funciones

El diseño descendente resuelve un problema efectuando descomposiciones en otros problemas más sencillos a través de distintos niveles de refinamiento. La programación modular consiste en resolver de forma independiente los subproblemas resultantes de una descomposición. El resultado de dividir reiteradas veces un problema en problemas más pequeños es una estructura jerárquica o en árbol.



Ejercicio Resuelto 1. **Realiza un programa en C que permita convertir grados Celsius a Fahrenheit y viceversa. El programa debe mostrar un menú para poder seleccionar qué opción se desea (más la opción terminar). Y preguntar la temperatura que desea convertir.**

Cada una de las conversiones se debe realizar por medio de un módulo.

La conversión de grados Celsius a grados Fahrenheit se obtiene multiplicando la temperatura en Celsius por 1,8 y sumando 32.

La conversión de grados Fahrenheit a grados Celsius se obtiene restándole 32 a la temperatura en grados Fahrenheit y dividiéndolo por 1,8.

Ejemplo de ejecución:

```
Seleccione qué tipo de conversión desea realizar:
1. Convertir grados Celsius a Fahrenheit.
2. Convertir grados Fahrenheit a Celsius.
3. Terminar el programa

Opción:1

Introduzca la temperatura en grados Celsius: 30
La temperatura en grados Fahrenheit es: 86
```

Solución:

```
#include<iostream>
using namespace std;

//Declaración de módulos
void mostrar_menu();
float Celsius_to_Fahrenheit(float);
float Fahrenheit_to_Celsius(float);

//Programa princpal
main(){

    //Declaración de variables
    int opcion;
    float celsius, fahr;

    do{
      mostrar_menu();
      cout<<"Opción: ";
      cin>>opcion;
      cout<<endl;

      switch(opcion){
            case 1: cout<<"Introduzca la temperatura en grados Celsius: ";
                  cin>>celsius;
                  fahr = Celsius_to_Fahrenheit(celsius);
                  cout<<"La temperatura en grados Fahrenheit es: "<<fahr;
                  break;
            case 2: cout<<"Introduzca la temperatura en grados Fahrenheit: ";
                  cin>>fahr;
                  celsius = Fahrenheit_to_Celsius(fahr);
                  cout<<"La temperatura en grados Celsius es: "<<celsius;
                  break;
            case 3: cout<<"Ha seleccionado la opción Salir."<<endl;
                  break;
            default: cout<<"Debe introducir una opción de 1 a 3"<<endl;
      }
```

```
      cout<<endl;
      cin.get();
      cout<<"Pulse Intro para continuar"<<endl;
      cin.get();

    }while(opcion!=3);

}

//Módulo que muestra el menú
void mostrar_menu()
{
    cout<<"Seleccione qué tipo de conversión desea realizar:"<<endl;
    cout<<"1. Convertir grados Celsius a Fahrenheit."<<endl;
    cout<<"2. Convertir grados Fahrenheit a Celsius."<<endl;
    cout<<"3. Terminar el programa."<<endl;
}

//Módulo que convierte de Celsius a Fahrenheit
float Celsius_to_Fahrenheit(float cel)
{
    float fah;

    fah = cel * 1.8 + 32;

    return(fah);
}

//Módulo que convierte de Fahrenheit a Celsius
float Fahrenheit_to_Celsius(float fah)
{
  float cel;

  cel = (fah-32)/1.8;

  return (cel);
}
```

Ejercicio Resuelto 2. **Realiza un programa que muestre un menú en el que a partir de un número entero n se elegirá una de las siguientes operaciones**:

- Opción 1: calcular de cuántas cifras se compone. (No contar los ceros a la izquierda)
- Opción 2: mostrar la cifra i-ésima de dicho número. La posición i debe pedirse al usuario.
- Opción 3: Salir

Solución:

```
#include <iostream>
using namespace std;

const int  OPCION_SALIR = 3;

int  CantidadCifras(int); //Declaración de la función
int  Cifra_i(int, int);
void menu();
int  Leer_Opcion();

int main(void)
{
   int opcion, n, resultado, i;

   do {
     menu();
     opcion = Leer_Opcion();
     if (opcion != OPCION_SALIR) {
```

```
       cout << "Introduzca un número: ";
       cin >> n;
     }
     switch(opcion) {
       case 1:
              resultado=CantidadCifras(n);
              cout <<"El número de cifras de "<<n<<" es "<<resultado<<endl;
              break;
       case 2:
              cout <<"Introduce la i: ";
              cin >>i;
              resultado=Cifra_i(n,i);
              cout << "La cifra "<<i<<" de "<<n<<" es "<<resultado<<endl;
              break;
       case OPCION_SALIR:
              cout << "FIN DEL PROGRAMA"<<endl;
     }
   } while(opcion != OPCION_SALIR);
}

int CantidadCifras(int m)
{
   int contador=1;

   while(m >=10) {
     m=m/10;
     contador ++;
   }
   return(contador);
}

int Cifra_i(int m,int i)
{
   int contador=1,cifra=0,devolver;

   cifra=m%10;
   while(m >=10 && contador<i) {
     m=m/10;
     cifra=m%10;
     contador ++;
   }
   if (contador==i)
     devolver=cifra;
   else
     devolver=-1;
   return(devolver);
}
void menu()
{
   cout << "1. Calcular el número de cifras de un número"<<endl;
   cout << "2. Mostrar la cifra i-ésima de un número"<<endl;
   cout << "3. Mostrar el reverso de un número"<<endl;
   cout << "4. SALIR"<<endl;
}

int Leer_Opcion()
{
   int opcion;
   do {
      cout << "Introduzca la opción: ";
      cin >> opcion;
   } while (opcion < 1  ||  opcion > 3);
   return(opcion);
}
```

Ejercicio Resuelto 3. **Dados los siguientes algoritmos**:

Algoritmo A	**Algoritmo B**

```
#include <iostream>
using namespace std;

void numMayor(int , int );


int mayor; //Declaración de variable global

int main(void) {
   int num1, num2;

   mayor = 0;
   cout << "Introduce dos números: ";
   cin >> num1 >> num2;

   numMayor(num1,num2);
   cout << "El mayor es "<<mayor<<endl;
}


void  numMayor(int n1, int n2)
{
    if (n1 > n2)
       mayor = n1;
    else
       mayor = n2;
}
```

```
#include <iostream>
using namespace std;

void numMayor(int , int );


int resultado;  //Declaración de variable global

int main(void) {
   int num1, num2;

   resultado = 0;
   cout << "Introduce dos números: ";
   cin >> num1 >> num2;

   resultado = num1 + num2 ;
   numMayor(num1,num2);
   cout << "Al sumar num1 y num2 da "<<resultado<<endl;
}


void numMayor(int n1, int n2)
{
    if (n1 > n2)
       resultado = n1;
    else
       resultado = n2;
}
```

a. ¿Qué ocurre? ¿Funcionan correctamente? Realiza una traza de los algoritmos anteriores para comprobar su funcionamiento.

La traza del algoritmo a:

	Programa Principal			procedimiento	
	num1	num2	mayor	n1	n2
	-	-	0		
supuesta la entrada:	4	6		4	6
			6		

llevaría a la ejecución:

Introduce un número : 4
Introduce un número : 6
El mayor es 6

En este programa desde el procedimiento numMayor *se está modificando una variable global (*mayor*) sin que se hubiese pasado como parámetro a dicho procedimiento, aunque se produzca un resultado correcto podría haberse producido un efecto lateral.* ***Esto es incorrecto****.*

La traza del algoritmo b:

	Programa Principal			procedimiento	
	num1	num2	resultado	n1	n2
	-	-	0		
supuesta la entrada:	4	6			
			10		
				4	6
			6		

llevaría a la ejecución:

Introduce un número : 4

Introduce un número : 6

El resultado de la suma de estos números es 6

En este programa desde el procedimiento numMayor *se está modificando una variable global sin que se hubiese pasado como parámetro a dicho procedimiento, como resultado se ha modificado la variable* resultado *que contenía un valor anteriormente (el de la suma de dos números) y por tanto se visualiza por pantalla un valor incorrecto . A esto se le denomina* ***efecto lateral****. Esto es incorrecto.*

b. ¿Crees que deberíamos modificar los algoritmos anteriores? ¿Por qué? ¿Cómo?

Si, en los dos aunque sólo uno produzca un resultado incorrecto. Deberíamos pasar como ***parámetro por referencia*** *una variable que almacenase el número mayor tras la ejecución del subalgoritmo para evitar efectos laterales.*

Algoritmo A (función devuelve valor)

```
#include <iostream>

using namespace std;

int numMayor(int, int);

int main(void) {
   int num1, num2, mayor;

   mayor = 0;
   cout << "Introduce dos números: ";
   cin >> num1 >> num2;

   mayor = numMayor(num1,num2);
   cout << "El mayor es "<<mayor<<endl;
}


// La función devuelve un número entero
int numMayor(int n1, int n2)
{
    int max;

    if (n1 > n2)
       max = n1;
    else
       max = n2;

    return(max);
}
```

Algoritmo B (paso por referencia)

```
#include <iostream>

using namespace std;

void numMayor(int, int, int &);


int main(void) {
   int num1, num2, mayor, resultado;

   resultado = 0;
   cout << "Introduce dos números: ";
   cin >> num1 >> num2;

   resultado = num1 + num2 ;
   numMayor(num1,num2,mayor);
   cout << "Al sumar num1 y num2 da "<<resultado<<endl;
}

//El procedimiento tiene un parámetro pasado por
//referencia
void numMayor(int n1, int n2, int &resultado)
{
    if (n1 > n2)
       resultado = n1;
    else
       resultado = n2;
}
```

Ejercicio Propuesto 1. Realiza un programa en C que calcule el área de diferentes figuras geométricas. Concretamente de un rectángulo, un triángulo y un círculo. El programa deberá mostrar un menú con las diferentes opciones (más la opción de terminar) y posteriormente preguntar los datos necesarios para el cálculo del área.

El área de un rectángulo se calcula como:

```
area_rectangulo = base * altura
```

El área de un triángulo se calcula como:

```
area_triangulo = (base * altura)/2
```

El área de un círculo se calcula como:

```
area_circulo = pi * radio^2
```

Ejemplo de ejecución:

```
Seleccione la opción deseada:
1. Área de un rectángulo
2. Área de un triángulo
3. Área de un círculo

Opción: 1

¿Cuál es la base? 4
¿Cuál es la altura? 5
El área del rectángulo es: 20
```

Ejercicio Propuesto 2. **Realiza un programa que muestre un menú al usuario que permita elegir entre dibujar un triángulo o un rectángulo formado por el carácter '*'. El programa debe tener tres módulos además del main():**

- **menu(): debe mostrar el menú con las opciones y validar la opción introducida. Es decir, si el usuario introduce una opción incorrecta debe mostrar un mensaje de error y volver a pedir la opción.**
- **triangulo(tam): dibuja un triángulo de tamaño *tam*.**
- **rectangulo(alto, ancho): dibuja un rectángulo de *ancho* x *alto***

Ejercicio Propuesto 3. **Modifica el programa anterior de tal forma que en lugar de pedir números al usuario, esos números se generen de forma aleatoria entre 1 y 20.**

Cómo generar números aleatorios en C

- Necesitaréis poner al principio del programa

```
#include <ctime>
#include <cstdlib>
```

- La primera instrucción dentro del `main()` debe ser `srand(time(0));` para inicializar el generador de números aleatorios. Esta instrucción solo se debe ejecutar una vez, no la metáis dentro de ningún bucle.
- Para generar un entero al azar entre *a* y *b* se usa `rand()%(b-a+1)+1`. Esto parece un poco complicado, pero para los casos más habituales es sencillo. Por ejemplo, para generar un número entre 1 y 20 se usaría `rand()%(20-1+1)+1`. O sea, simplemente `rand()%20+1` .Esto devuelve un valor que podéis asignar a una variable o devolver como resultado de una función, por ejemplo

```
int valor = rand()%20+1;
```

Ejercicio Propuesto 4. **Debes implementar un programa en C que permita jugar al juego Piedra, Papel, Tijera. El juego constará de varias rondas y ganará el que gane 2 rondas. El juego terminará cuando el usuario ya no quiera seguir jugando.**

El programa deberá mostrar en cada ronda, las rondas que lleva ganadas cada jugador. Y al final de la partida, las partidas que lleva ganadas cada jugador.

Módulos:

Para implementar el juego, además de la función main(), el programa debe contener los siguientes módulos:

•Módulo "apuestaUsuario". Este módulo debe pedir al usuario si quiere Piedra, Papel o Tijera. Se tendrá en cuenta si comprueba que la elección del usuario es correcta. El módulo debe mostrar la apuesta del usuario y debe devolver la elección del usuario (Piedra (p), Papel(a) o Tijera(t)).

•Módulo “piedraPapelTijera”. Este módulo mostrará y devolverá si la máquina ha elegido Piedra, Papel o Tijera. Debe calcularlo de manera aleatoria, para ello debe utilizar la función rand(), que genera un número aleatorio.

•Módulo “comprobar”. Este módulo comprobará cuál de los dos jugadores ha ganado la ronda y mostrará los resultados. También comprobará si alguno de los dos jugadores ha ganado ya 2 rondas, y por tanto, la partida. Si, es así, mostrará quién ha ganado y las partidas que lleva ganadas cada jugador.

Ejemplo de ejecución básico:

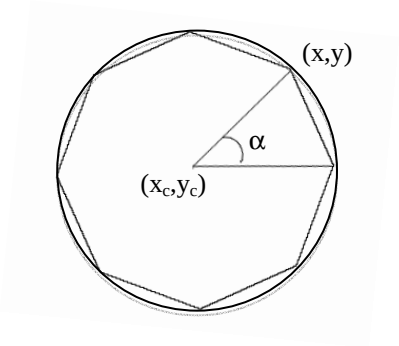
```
*** PIEDRA, PAPEL O TIJERA ***
¿Quieres jugar? (s/n) s
¿Piedra, papel, tijera? (p/a/t) p
Papel
Gana la máquina.
Rondas ganadas:
Maquina: 1
Usuario: 0

¿Piedra, papel, tijera? (p/a/t) t
Papel
Tú ganas.
Rondas ganadas:
Maquina: 1
Usuario: 1

¿Piedra, papel, tijera? (p/a/t) a
Tijera
Tú ganas.
Gana la máquina.
Rondas ganadas:
Maquina: 2
Usuario: 1

¡Oh! ¡Has perdido!
Partidas ganadas:
Maquina: 1
Usuario: 0

¿Quieres jugar? (s/n) n
Juego terminado
```

Ejercicio Propuesto 5. **Escribe un programa en C que dibuje un polígono regular empleando la librería gráfica `gfx`. El programa solicitará las coordenadas (x_c, y_c) del centro del polígono y su radio r, así como el número de lados a dibujar. Una vez comprobado que no se excede de la ventana gráfica, el programa dibujará el polígono centrado en la posición (x_c,y_c) .**

Para hallar los vértices del polígono puedes utilizar las siguientes ecuaciones:

$$x = x_c + r \cdot \cos(\alpha)$$
$$y = y_c + r \cdot \sin(\alpha)$$

donde α es el ángulo correspondiente al vértice, como se observa en la figura.



Puedes hacer uso de la rutina `gfx_line()` para dibujar los lados del polígono.

El programa debe tener como mínimo los siguientes módulos:

- **`validaCoordenadas`: módulo que comprueba si algún vértice del polígono se sale de la zona de trazado. El programa debe dibujar el polígono sólo si todos los vértices se encuentran en la zona de trazado**
- **`validalados`: función que se encarga de pedir el número de lados del polígono y validar que se encuentra entre 4 y 15. Esta función debe obligar al usuario a introducir un número correcto.**

Ejemplo de polígono de 8 lados y radio 90

ERRORES DE PROGRAMACIÓN COMUNES

- **Utilizar punto y coma en la definición de una función.** Cuando se define una función , su cabecera no termina con punto y coma. El punto y coma se utiliza en las declaraciones, es decir, en los prototipos de las funciones. Lo mismo se puede decir si se trata de un procedimiento (función tipo void en C).
- **Llamar a una función con un número diferente de parámetros.** Hay que asegurarse de que se pasa el número exacto de parámetros cuando se invoca a una función o a un procedimiento.
- **Devolver un valor que no coincide con el tipo de la función.** Cuando se devuelve un valor utilizando *return*, hay que asegurarse de que el resultado de la expresión situada dentro de *return* coincide con el tipo de la función.
- **No utilizar *return* en una función que devuelve un valor.** Cuando una función devuelve un valor, hay que utilizar *return* para devolverlo.
- **Pasar un parámetro real con distinto tipo al parámetro formal.** Cuando llame a una función o a un procedimiento, el tipo de los parámetros reales debe coincidir con el tipo de los correspondientes parámetros formales.
- **Olvidar los paréntesis en una función que no acepta parámetros.** Aunque una función no acepte parámetros, cuando se realiza la llamada a la misma deben utilizarse paréntesis, sin nada entre ellos. Si no se colocan los paréntesis, se obtiene un error de compilación.
- **Declarar un prototipo que no coincide con la definición de la función.** Cuando se define una función, hay que asegurarse de que tanto la definición como su declaración tienen el mismo prototipo, es decir, devuelven el mismo tipo y aceptan el mismo número y tipo de parámetros.